**Insert Operation:** 698.130 milliseconds

**Append Operation:** 1.273 milliseconds

| Array Size | Append Time (ms) | Insert Time (ms) |
| --- | --- | --- |
| 10 | 0.034041999999999996 ms | 0.014875 ms |
| 100 | 0.005167 ms | 0.007417 ms |
| 1,000 | 0.026125 ms | 0.089667 ms |
| 10,000 | 0.264334 ms | 5.151375 ms |
| 100,000 | 0.6551669999999999 ms | 702.9951669999999 ms |

**Findings:**

The results from testing doublerAppend and doublerInsert with arrays of different sizes show that doublerAppend scales much better. As the array size increases from 10 to 100,000 elements, the time it takes for doublerAppend to execute increases only slightly, from about 0.034 ms to 0.655 ms. This small increase suggests it handles large arrays efficiently. On the other hand, doublerInsert becomes significantly slower with larger arrays, especially noticeable when it jumps from 5.151 ms for 10,000 elements to a massive 702.995 ms for 100,000 elements. This indicates that inserting at the beginning of an array, which requires shifting all other elements each time, is much less efficient as the array grows. Therefore, doublerAppend is clearly the better choice for operations involving large datasets because it remains fast and stable even as array sizes increase.

**Why doubleInsert is slower:**

The doublerInsert function is slower because it inserts numbers at the beginning of the array, which is an inefficient process in JavaScript. Each time a number is inserted, all the other numbers in the array have to be moved one spot to make room. This movement is time-consuming because it has to be done every single time a new number is added. As the array gets bigger, more numbers need to be moved, so the process takes longer. In contrast, adding numbers to the end of the array with the doublerAppend function doesn't require moving other numbers around, which is why it's much faster and doesn't slow down as much as the array grows.